



國立臺北科技大學

資訊工程系碩士班

碩士學位論文

**RobotFramework-ParallelAppiumLib 平行  
手機測試函式庫**

**RobotFramework-ParallelAppiumLib  
Parallel Mobile App testing Library**

研究生：張洋洋

指導教授：鄭有進教授、謝金雲教授

中華民國一百零七年七月

# 摘要

論文名稱：RobotFramework-ParallelAppiumLib 平行手機測試函式庫

頁數：三十八頁

校所別：國立臺北科技大學 資訊工程 研究所

畢業時間：一百零六 學年度 第二學期

學位：碩士

研究生：張洋洋

指導教授：鄭有進教授、謝金雲教授

關鍵詞：平行化測試、手機測試、跨平台

市面上諸多 App 具有聊天、推播功能且多種不同平台如 iOS、Android、WindowPhone 等上同時發表。在開發 App 的過程中，時常需要透過 Automated Test 來維護品質，但是在相異平台間要測試聊天、推播這種涉及多支手機即時通訊的功能，在多種 OS 需要各種不同測試工具的狀況之下，變得難以測試，而且測試難以自動化。

為了能夠在各種 OS 下自動化測試多機即時互動的功能，我們需要一個新的測試工具。建構在 Robot Framework 自動化測試之下，運用 Appium 能在所有平台上運行的特性，透過本專案－Robotframework Parallel Appium Library 運行並控制多個 Appium server，再加上 RobotFramework-runKeywordAsync Library，開啟多個執行緒，讓每個 Appium Server 與 Client 平行運行，進而達到異平台多機平行自動化測試的測試效果。

# ABSTRACT

Title : RobotFramework-Parallel Appium Lib: Robot Framework parallel mobile app testing

Pages : 38

School : National Taipei University of Technology

Department : Computer Science and Information Engineering

Time : August, 2018

Degree : Master

Researcher : Yang-Yang Chang

Advisor : Yu Chin Cheng Ph.D, Chin-Yun Hsieh Ph.D

Keyword : Android, iOS, Appium, parallelization, Automated Test, multi-platform

There is great amount of Apps that feature chatting, file transferring and other interactive functions. Also, most of the App mentioned above support multiple mobile platforms such as iOS, Android and WindowPhone. While in development, qualities of software should be constantly test through Automated Tests and Acceptance Tests. But creating Automated Tests and Acceptance Tests for interaction features on multiple platforms are quite difficult due to the difference in requirements between platforms. Different platform requires different tools, environment etc.

In order to make production of Automated Tests and Acceptance Tests on interaction features more efficient and easy, we need to create a new App testing tool. By combining Robot Framework's keyword driven feature, Appium's ability to run tests on most mobile platforms, and RobotFramework-RunKeywordAsync's multithreading potential, we can create a tool that runs and controls multiple Appium servers and clients simultaneously, thus achieving the goal – easier test creation for interaction features.

# 目 錄

摘 要 i

ABSTRACT..... ii

誌 謝 ..... 錯誤! 未定義書籤。

目 錄 ..... iii

表目錄 ..... vi

圖目錄 ..... vii

第一章 緒論..... 1

1.1 研究背景與動機..... 1

1.2 研究目標..... 2

第二章 背景知識與相關技術..... 3

2.1 Appium ..... 3

2.2 Robot Framework ..... 5

2.3 Robot Framework -Run keyword Async ..... 6

2.4 Robot Framework –Appium Library ..... 7

第三章 Appium 常用方式分析與設計 ..... 8

3.1 Appium 使用方式 ..... 8

3.1.1 安裝與設定 Appium 環境 ..... 8

3.1.2 運行與設定 Appium 伺服器 ..... 8

3.1.3 啟動 Session ..... 10

3.1.4 撰寫測試項目 ..... 11

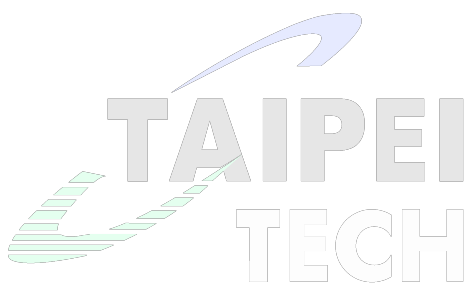
3.2 需求分析..... 12

3.2.1 對多支實體手機進行操作..... 12

3.2.2 降低使用門檻..... 12

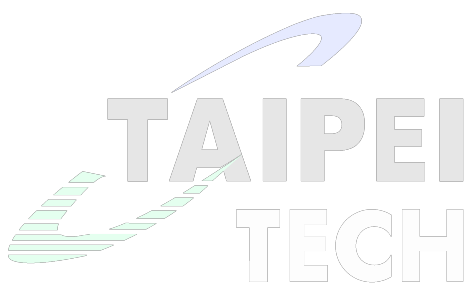
3.2.3	平行測試與多支實體手機互動.....	12
3.3	開發新工具 – Parallel Appium Library .....	14
3.3.1	對多支實體手機進行操作.....	14
3.3.2	降低使用門檻.....	14
3.3.3	平行測試與多支實體手機互動.....	14
第四章	系統架構分析與實作.....	16
4.1	Parallel Appium Library 架構設計 .....	16
4.2	AppiumInstance 管理 .....	19
4.2.1	參數設定.....	19
4.3	AppiumInstance .....	20
4.3.1	AppiumInstance 參數與參數解析 .....	20
4.3.2	預設 Desired capabilities 與其他參數.....	22
4.3.3	測試指令 keywords.....	22
4.3.4	Parallel Appium Lib 使用實例.....	27
第五章	比較.....	30
5.1	比較內容.....	30
5.2	環境介紹.....	30
5.3	功能比較.....	31
5.3.1	Appium Library 功能 .....	31
5.3.2	ParallelAppiumLib 功能.....	32
5.3.3	比較結果.....	32
5.4	腳本撰寫方式.....	32
5.4.1	原始 Appium .....	32
5.4.2	ParallelAppiumLib 撰寫方式.....	34
5.4.3	比較結果.....	34

第六章 結論與未來研究方向.....	35
6.1 結論.....	35
6.2 未來展望.....	35
參考文獻.....	37



## 表目錄

表 3.1	功能與解決問題整理.....	15
表 4.1	Parallel Appium Lib 環境需求.....	16
表 4.2	addInstance 參數內部用圖表 .....	21
表 4.3	FindElement 系列 keyword.....	23
表 4.4	Click 系列 keyword.....	24
表 4.5	sendKeys 系列 keyword.....	25
表 4.6	wait 系列 keyword.....	26
表 5.1	電腦環境.....	30
表 5.2	手機環境.....	30



# 圖目錄

圖 2.1	Appium 結構示意圖 .....	4
圖 2.2	Robot Framework 模組結構圖 .....	6
圖 2.3	Run Keyword Async .....	7
圖 3.1	使用 windows command line 啟動 Appium 伺服器 .....	9
圖 3.2	用 desired capabilities 分別在測試 iOS 與 Android 時的範例 .....	10
圖 3.3	用 Accessibility ID 尋找物件 .....	11
圖 3.4	Robot Framework-Appium Library 測試範例 .....	11
圖 3.5	單執行續測試即時彈出視窗可能遇見的錯誤狀況.....	13
圖 4.1	Parallel Appium Lib 架構設計圖 .....	17
圖 4.2	ParallelAppiumLib class diagram (簡化).....	18
圖 4.3	設定並運行兩個伺服器的範例 .....	20
圖 4.4	以 find_element_by_accessibility_id 為例，AppiumInstance 運行測試的 Sequence Diagram .....	27
圖 4.5	ParallelAppium 兩隻手機聊天測試情境流程圖 .....	29
圖 5.1	switch_application 的使用範例 .....	31
圖 5.2	透過 Python-client 設定 desired_capabilities .....	33
圖 5.3	透過 python-client 尋找物件 .....	33
圖 5.4	ParallelAppiumLib 的使用範例.....	34



# 第一章 緒論

## 1.1 研究背景與動機

2017 Q2 市佔率前兩名的智慧型手機 OS—iOS[1] (12.1%)與 Android[2] (87.7%)平台，兩者市佔率總和已經到達 99.8%，對於許多開發者而言，為了觸及所有的使用者，同一個產品或服務需要在 iOS 與 Android 上分別開發出不同的 App。

如何縮短開發時間，同時維護軟體品質一直是開發者面對的重要挑戰與問題[3]。為了保障產品品質，開發者需要在開發週期中隨著更新持續對 App 進行自動化測試 (Automation Test[4]) 及驗收測試(Acceptance Test[5])。在現有工具之下，開發者必須在不同 OS 下分別開發 App，而同時在不同的系統下開發 App 的測試機制，增加維護測試的難度。以 iOS 與 Android 為例，iOS 下的 App 只能在蘋果電腦上測試，而 Android 可以在 Windows 與大部分 Linux 上建立測試環境。若要同時開發 iOS 與 Android 版本的 App，開發者必須在兩個系統上分別開發功能類似的程式，同時分別在兩種 OS 上建立不同的測試環境。

隨著網路通訊的基礎建設與技術的越趨發達，現在的 App 大多具有推播、聊天、視訊等與其他使用者即時互動的功能。若要測試這些功能，開發者需要同時運行多支手機才能更能接近現實生活中的情境。但是這種涉及多支手機即時通訊的功能，在多種手機 OS 需要各種不同測試工具的狀況之下，變得難以測試，而且測試難以自動化。

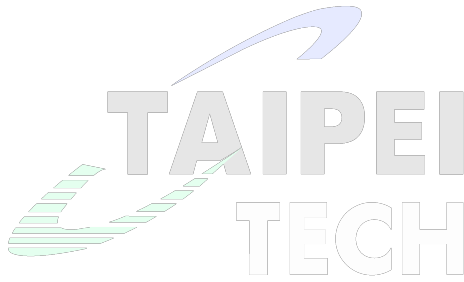
Appium[6]是一種開源自動化手機 App 測試工具，可以在虛擬機、實機上運行測試，同時具備測試 Native[7]、Hybrid[8]、Mobile Web[9] App 的能力，而 Appium 最大的特點在於它能夠測試 Android、iOS、Windows Phone 等行動裝置上的 App。但是 Appium 採用 Server(Appium server) – Client(腳本)的結構，而一組 Server-Client 只能在一支手機上運行腳本，現行方法大多為在伺服器之間切換，但單執行續下的切換會造成時間空檔，使得有些一閃即逝的情境無法測試。

Robot Framework[10]是一種通用的自動化測試平台，使用者可以透過各種 keyword library 達成各種測試功能，簡化測試自動化的流程。

## 1.2 研究目標

本論文受到劉展君[11]研究的啟發，的目標旨在於開發一組 Robot Framework 下的 keyword library。運用 Appium 能在通用於多平台的特性，用多執行緒的架構，達成在不同手機 OS 平行測試效果，使得多機互動的功能得以更貼近真實使用者情境進行測試，簡化多機互動功能的測試難度，解決單執行緒時間空檔的問題，同時也可用於測試平行化—增加整體測試效率。

最後本論文 – Robot Framework-ParallelAppiumLib 將開源至 Github，供社群利用。




## 第二章 背景知識與相關技術

### 2.1 Appium

Appium 是一套開源的手機 App 自動測試工具，可以跨平台測試各種平台的手機，各種形式的 App。使用者透過 Appium 測試手機不需要為了自動化測試而修改或重新編譯所開發的 App。目前 Appium 官方有提供各種程式語言的 Appium 客戶端，如 Ruby、Python、JAVA、JavaScript、Objective C、PHP、C#等，測試腳本可以用各種程式語言撰寫，不需要被鎖定在特定語言或框架之下。

Appium 使用 Android、iOS、WindowsPhone 各團隊開發的自動化套件，所以不需要為了測試而特別在修改要測試的 App，例如在 App 中嵌入程式碼、library 等。

Appium 所套用的原生自動化測試套件：

- 
- iOS 9.3+: Apple's XCUITest[12]
  - iOS 9.3 以前: Apple's UIAutomation[13]
  - Android 4.2+: Google's UiAutomator/UiAutomator2 [14]
  - Android 2.3+: Google's Instrumentation. (Selendroid) [15]
  - Windows: Microsoft WinAppDriver[16]

另外，Appium 腳本可以跨程式語言的特性源自於 server-client 的結構。Appium 建基於 WebDriver API 之上，server-client 透過 JSON Wire Protocol 溝通。所以只要能夠對 Appium 伺服器發送 RESTful[17] HTTP request 的語言，都可以成為 Appium 客戶端。

在開始執行測試的時候，客戶端會像伺服器傳送稱為 Desired Capabilities[18]的 JSON[19] object，藉著 Desired Capabilities，伺服器會建立一組 session，Appium 客戶端可以設定在這個 session 之中伺服器要運用哪些參數運行，譬如要測試的平台(Android、iOS...)，欲操作實體手機之 UUID，Log 紀錄的內容，還有伺服器所用之 port 號碼等。

而在測試之中，客戶端依照腳本對伺服器端傳送一連串的測試指令，達成自動化測試效果。

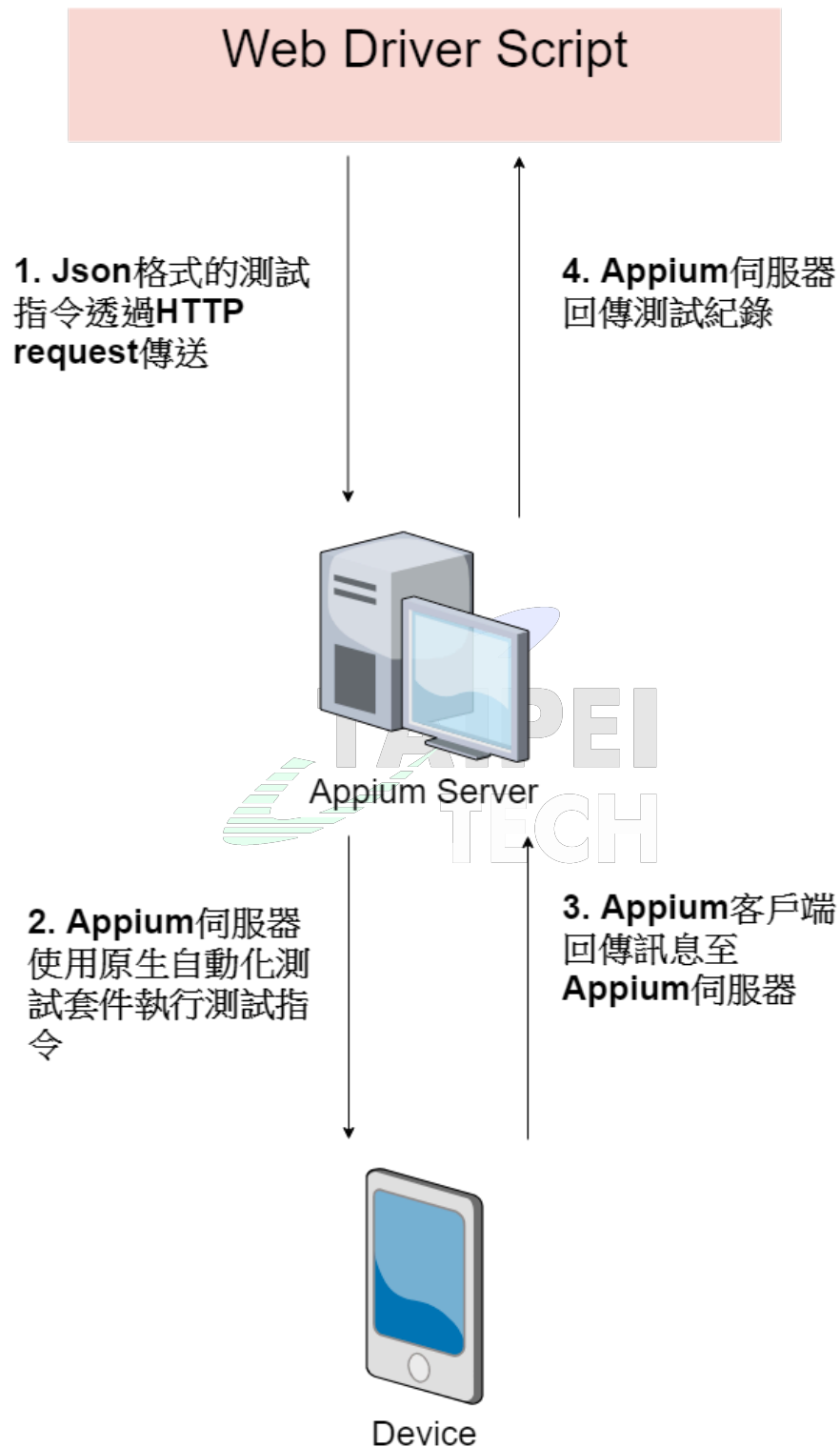
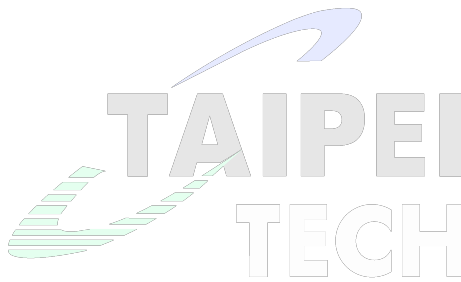


圖 2.1 Appium 結構示意圖[20]

## 2.2 Robot Framework

Robot Framework 是一個通用性很高，用 python 編寫的功能自動化測試軟體框架，通常用於運行驗收測試 (Acceptance Test) 或測試驅動的開發流程 (Test-Driven Development)。它的特點在於它表格式的腳本以及 Keyword-driven-testing[21]。測試功能可以透過套用由 JAVA 或 python 等語言所撰寫的 keyword library 來擴充，使得 Robot Framework 能做到更多更複雜的是事情。網路社群上有豐富的資源，許多第三方設計的 library 可以下載使用。綜合以上特色，使得 Robot Framework 變得非常熱門的測試工具。

在執行腳本的時候，Robot Framework 會先解析腳本資料，然後用自身或 library 擴充之 keyword，直接或透過測試工具與待測系統溝通。由於測試是透過 command line 運行，在測試完畢後，Robot Framework 會產出 XML 格式的 log 與報表，方便了解測試情況與結果。



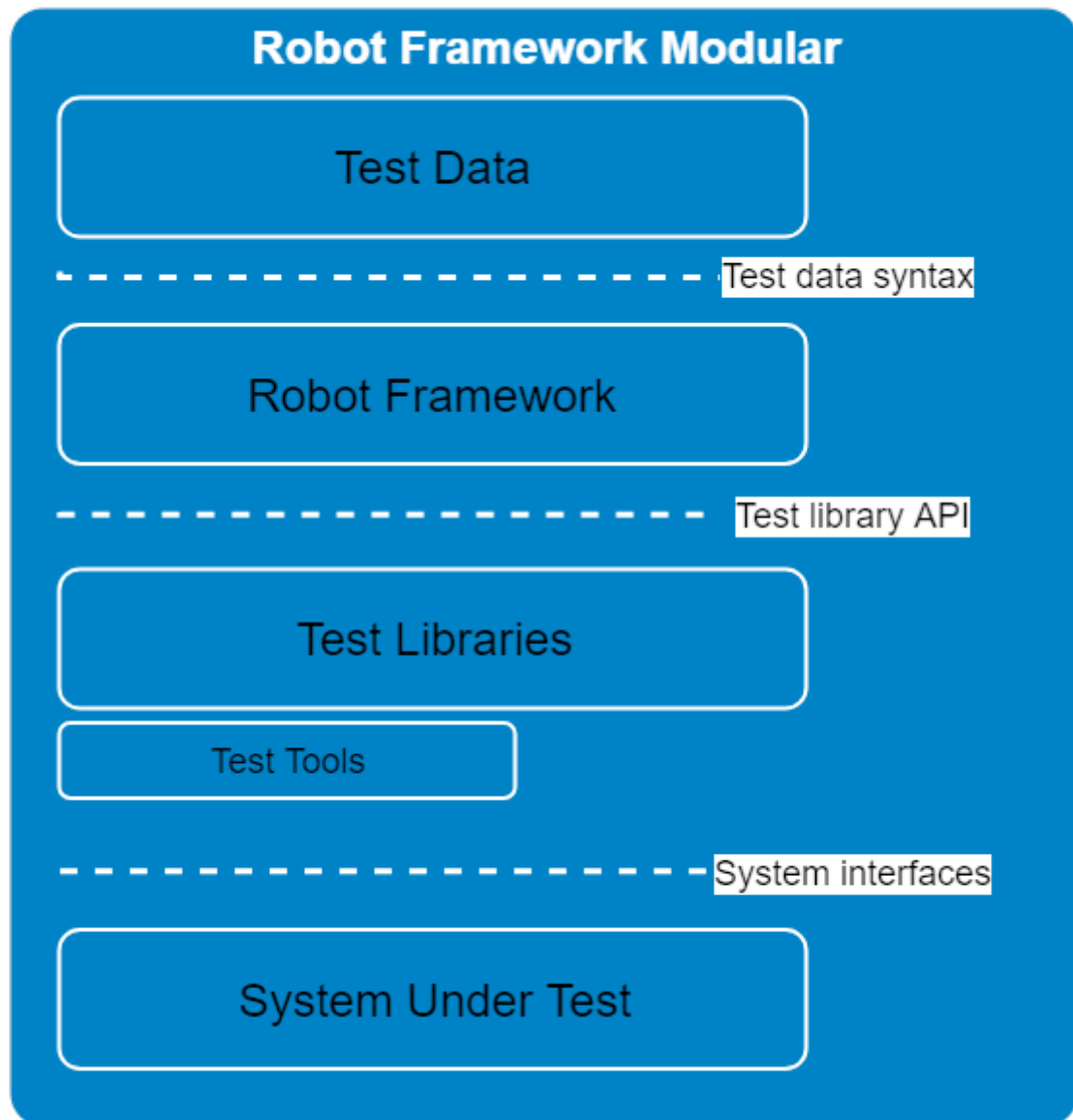


圖 2.2 Robot Framework 模組結構圖[22]

## 2.3 Robot Framework -Run keyword Async[23]

Robot Framework -run keyword async 是一個開源的 Robot Framework keyword library，具有開啟多執行緒的功能，使得 keyword 能分別在不同執行緒上同時運行。另外，可以設定 Timeout 參數，給予執行緒執行任務的時間限制。

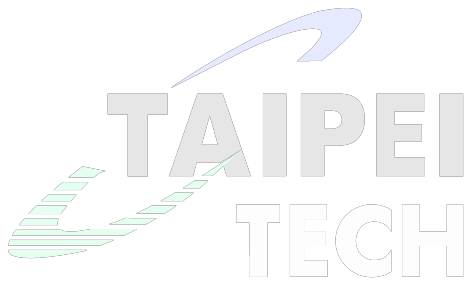
開啟執行緒範例：

*`${handle}= Run Keyword Async <keyword name> <first argument> <second argument>`*

圖 2.3 Run Keyword Async [24]

## 2.4 Robot Framework –Appium Library

Robot Framework –Appium Library 是一個 Robot Framework keyword library，使用者可以用他來對 Appium server 發送指令，將 Robot Framework 當成 Appium 客戶端，讓使用者的環境下撰寫 Appium 測試腳本。



## 第三章 Appium 常用方式分析與設計

本章會說明 Appium 與 Robot Framework-Appium Library[25]的架構與腳本撰寫方法，分析其所受到的限制，並提出具體的改善的方法。

### 3.1 Appium 使用方式

本節會描述 Appium 以及 Robot Framework – Appium Library，如何測試手機，而使用者平時是如何使用他們。

#### 3.1.1 安裝與設定 Appium 環境

安裝 Appium 有兩種常見的方法，一個是透過 Appium\_Desktop，二是透過 Node.js 下指令安裝。

安裝 Appium 最簡單的方法是自官方網站下載並安裝包裝好的 Appium\_Desktop\_installer，這樣不只安裝帶有 GUI 的 Appium 伺服器，還有 Appium inspector。Appium inspector，是一個 Appium 的輔助工具，可以協助使用者尋找定位所需要執行動作的物件，並實際遙控虛擬機或實機。

透過 Node.js 安裝，則需要先安裝 Node.js，安裝後再 command-line 下指令：

**– npm install -g appium**

Appium 還需要其他環境的支援，設定的細節將於第四章描述。

#### 3.1.2 運行與設定 Appium 伺服器

Appium 伺服器是由 Node.js[26]撰寫，安裝後透過 Command Line Tool(Windows)或 Terminal(OSX)直接以指令 “\$appium”即可執行，後面可以繼續輸入參數如：“--port 4723”設定伺服器使用 port 4723。

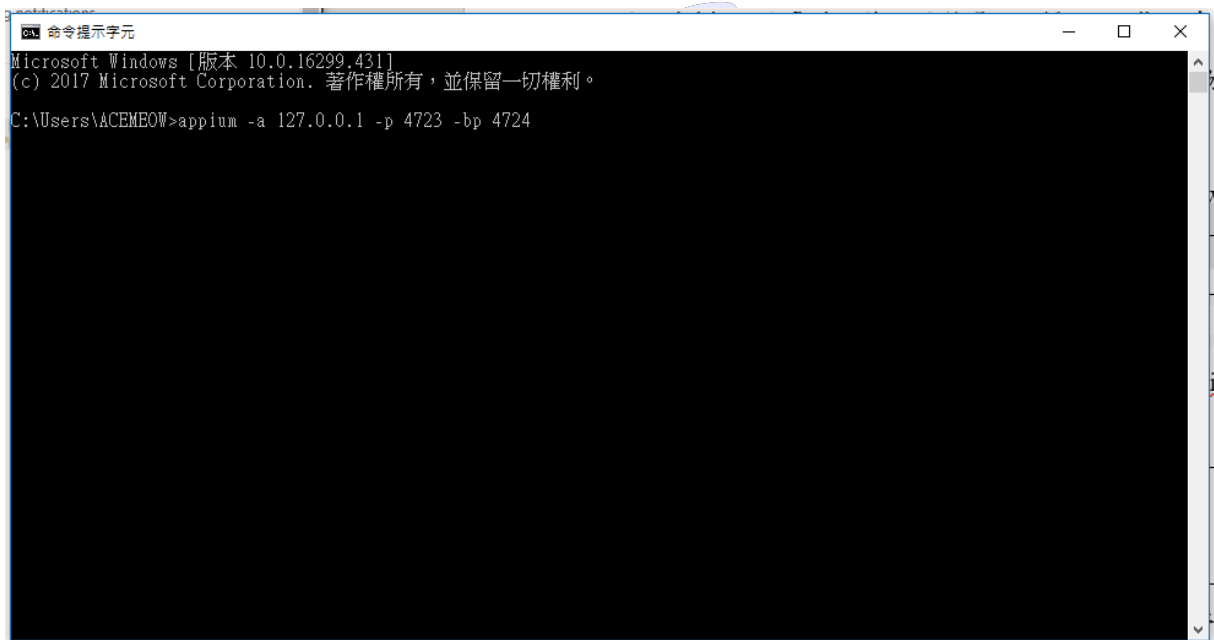
在本機啟動 Appium 伺服器的範例：

**Appium -a 127.0.0.1 -p 4723**



以下為一些常用的參數[27]：

指令	預設值	概述	範例
-a, --address	0.0.0.0	接收資訊的 IP 位置	--address 0.0.0.0
-p, --port	4723	接收資訊的 port	--port 4723
-g, --log	null	將 log 記錄存至檔案	--log /path/to/appium.log
--safari	false	(iOS 用)啟動 Safari 瀏覽器	
-bp, --bootstrap-port	4724	(Android 用)收聽 port	--bootstrap-port 4724



```
命令提示字元
Microsoft Windows [版本 10.0.16299.431]
(c) 2017 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\ACEMEOW>appium -a 127.0.0.1 -p 4723 -bp 4724
```

圖 3.1 使用 windows command line 啟動 Appium 伺服器

### 3.1.3 啟動 Session

市面上有數種工具與方法可以啟動、擔任 Appium 客戶端，啟動 Session 並撰寫 Appium 自動化測試，接下來我們會列舉幾個常用的工具分析，1. Python/JAVA 腳本 2. Robot Framework-Appium Library。

一般最常見的 Appium 腳本是以 Appium 團隊開發的 Python-client[28]或 JAVA-client[29]等撰寫。這些客戶端以 library 的形式包裝，使用者可以直接在 python/JAVA 程式碼呼叫 library 中的函式，而這些函式會向 desired capabilities 所指定之 Appium 伺服器發送 Request。Python-client 與 JAVA-client 之間只有語法不同，功能相仿，以下以 Python-client 為代表。

透過向伺服器傳送 desired capabilities，客戶端與伺服器之間依照 desired capabilities 所傳達的資料開建立一個 session，而在這 session 之中只有一個手機可以跟這個 server 溝通。簡而言之，在一個 session 之內，一組 Appium 的伺服器與客戶端只能測是一隻手機。

以下是再 python-client 之中用 desired capabilities 設定 session 簡單的例子：

```
# Android environment
import unittest
from appium import webdriver

desired_caps = {}
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '4.2'
desired_caps['deviceName'] = 'Android Emulator'
desired_caps['app'] = PATH('../..../apps/selendroid-test-app.apk')

self.driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
```

```
# iOS environment
import unittest
from appium import webdriver

desired_caps = {}
desired_caps['platformName'] = 'iOS'
desired_caps['platformVersion'] = '7.1'
desired_caps['deviceName'] = 'iPhone Simulator'
desired_caps['app'] = PATH('../..../apps/UICatalog.app.zip')

self.driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
```

圖 3.2 用 desired capabilities 分別在測試 iOS 與 Android 時的範例

另外在 Robot framework -Appium Library 中，使用者可以透過 keyword “Open Application”來輸入 desired capabilities 並啟動一個 session，以下是 Open Application 的使用範例：

<b>Open Application</b>	http://localhost:4723	platformName=Android	platformVersion=4.2	deviceName=XYZ0001234567890
app=test.apk	appPackage=com.test.demo	appActivity=MainActivity		

### 3.1.4 撰寫測試項目

在傳送 desired capabilities、成功建立 session 後就可以開始遙控手機進行測試，python-client 提供各種搜尋物件的方法，如：xpath、class、ID、Accessibility ID 等，還有動作指令 click、swipe、輸入字串等。下面的範例示範用 python-client 以 accessibility\_id 尋找物件。

```
el = self.driver.find_element_by_accessibility_id('Animation')
self.assertIsNotNone(el)
```

圖 3.3 用 Accessibility ID 尋找物件

另外使用者也可以運用 Robot Framework-Appium Library 撰寫測試腳本。Appium Library 提供各種搜尋、點擊、輸入字串等功能 keyword。除此之外，Robot Framework-Appium Library 還有提供切換伺服器的 keyword，讓使用者可以在單執行緒的條件下測試兩隻手機。以下是用 Robot Framework-Appium Library 撰寫測試的範例：

```
Click Element    accessibility_id=Add Contact
Input Text      id=com.example.android.contactmanager:id/contactNameEditText    ${contact_name}
Input Text      id=com.example.android.contactmanager:id/contactPhoneEditText    ${contact_phone}
Input Text      id=com.example.android.contactmanager:id/contactEmailEditText    ${contact_email}
Click Element    accessibility_id=Save
```

圖 3.4 Robot Framework-Appium Library 測試範例

## 3.2 需求分析

本節將依據 3.1 節中敘述的現況進行分析，說明其可改進的項目，並具體描述改善之需求。

### 3.2.1 對多支實體手機進行操作

根據 3.1.1 節中所提到，當使用者透過 Appium-client 送出 desired capabilities 後，伺服器只能夠跟一個手機溝通。這是由於一個伺服器對一隻手機是 Appium 基本結構上的設計，若需要克服這個問題，我們無法改變 Appium 本身，而需要另外建立一個系統同時運行並管理多個 Appium 伺服器，才能夠同時對多支手機進行操作。

### 3.2.2 降低使用門檻

在 3.1.1 有提到，如果用 python 或 JAVA 撰寫測試腳本會有學習門檻，並非所有人都会寫程式，也不是所有會寫程式的人都會 python 或 JAVA。我們需要一種方法，降低學習門檻，增加系統的親民度。

### 3.2.3 平行測試與多支實體手機互動

線上即時聊天功能在真實使用者情境之中，涉及多支手機在分開的系統上同時運行。為了要模仿真實使用情境，手機之間應該是分開平行運作的，但是 Appium 自身伺服器與客戶端的結構不允許同時多支手機同時運行，更無法平行運作。如果用 Robot Framework-Appium Library，雖然他能夠透過切換的方式切換伺服器，但是兩隻手機的測試腳本不是完全分開的兩個系統，而是在單執行續上互相切換。

若要更貼近真實使用情境，客戶端應該也要是平行運行的。

### 3.2.4 即時反應

線上戶動的過程中常有一閃而逝的彈出式訊息、提示，又或是小遊戲需要使用者立即反應介面內容。若使用單執行續的進行，在單執行續切換測試手機的過程中，需要驗證的狀態已經消失。

若使用多執行續持續追蹤手機的介面的狀態，我們便可以避免在多機測試中，這些即時訊息無法被測試、驗證的情形。

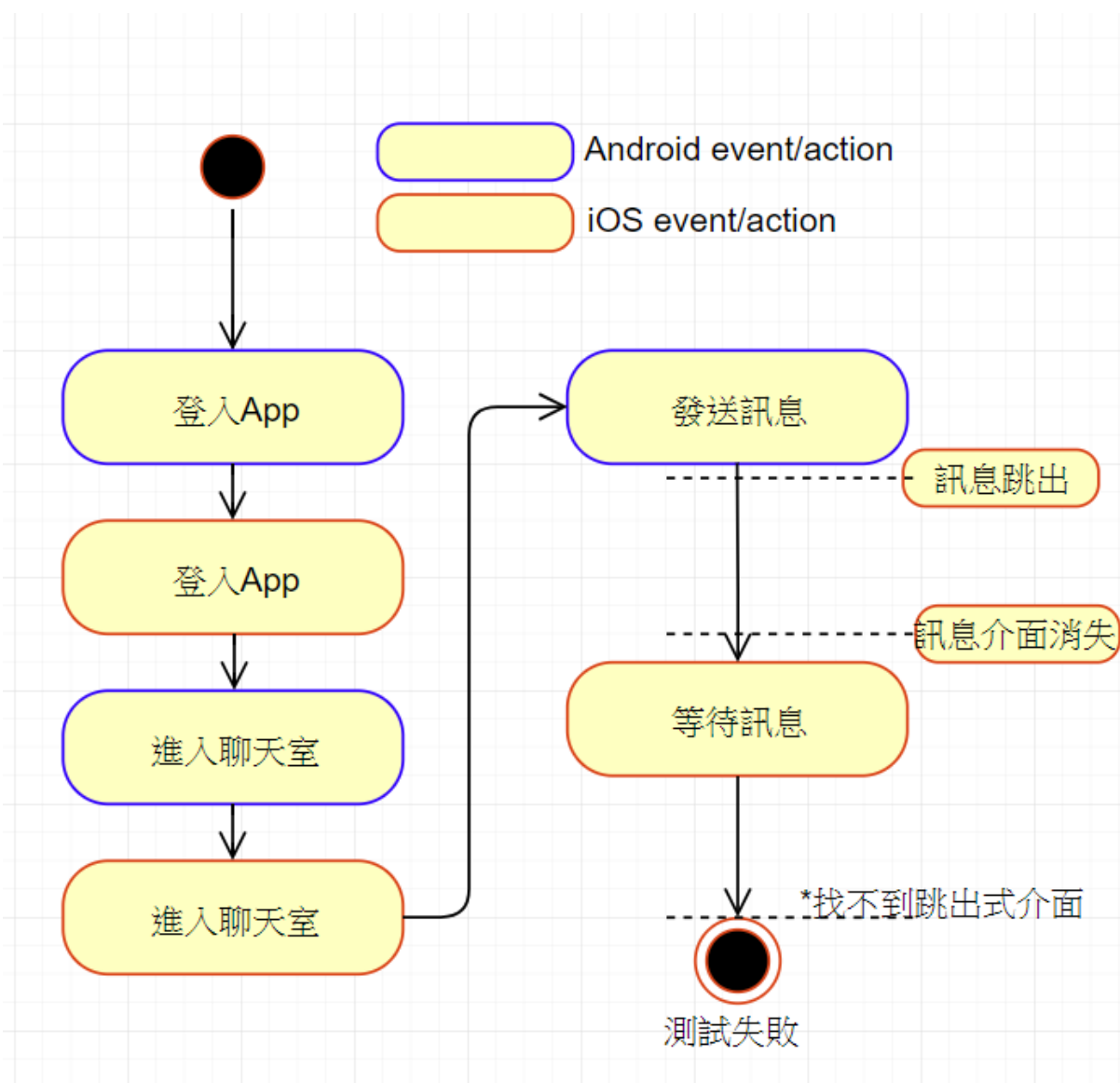


圖 3.5 單執行續測試即時彈出視窗可能遇見的錯誤狀況

## 3.3 開發新工具 – Parallel Appium Library

本節會依照 3.2 節所提出的需求，提出對應的解決方法，改善 Appium 與 Robot Framework-Appium Library 針對多機平行測試不完善的部分。

### 3.3.1 對多支實體手機進行操作

在 3.2.1 節中所提到的問題，一個伺服器在一個 session 中只能操作一隻手機，而我們能做的是同時運行多個伺服器。為了要同時操作多支手機，我們可以再 python 之中用多執行續呼叫 command line，接受使用者所需要的伺服器參數啟動多個 Appium 伺服器。這樣我們就可以同時有多支手機可以操作。

### 3.3.2 降低使用門檻

在 3.2.2 中有提到，若用 python/JAVA 撰寫測試腳本會有學習門檻。為了解決這個問題，我們可以運用 Robot Framework，運用其表格化的腳本結構與 keyword driven 的特質，不只使撰寫腳本更加容易上手，降低使用門檻，同時也增加了撰寫測試腳本的方便度。

### 3.3.3 平行測試與多支實體手機互動

在 3.2.3 節中所提到，為了更貼近使用者情境的測試類似即時聊天的功能，每一隻參與測試的手機在測試時應該分開獨立的運行，解決 Robot Framework-Appium 不支援多執行續的問題。

由於我們的新工具會再 Robot Framework 下運行，客戶端的平行問題我們可以運用 Robot Framework – Run keyword async 來分別運行對應伺服器數量的客戶端。同時客戶端必須帶有測試手機會運用到的 keyword，如點擊、輸入字串、swipe 等動作。

綜合以上數點需求，在新的工具之中，我們可以將 Appium 伺服器、客戶端與手機一組視為一個個體來操作。並用平行的方法控制每一組，達成多機平行測試的效果。

同時，因為在多執行續下，系統可以持續追蹤手機介面的狀態，因此便能解決一閃而逝的彈出式訊息無法被驗證的問題。

表 3.1 功能與解決問題整理

Parallel Appium Library 應有的特色	解決的問題
Parallel Appium Library 應能同時啟動並操縱多支手機	解決 Appium 不能同時操作多支手機的問題，同時這也是平行測試的基礎。 解決即時彈出式訊息無法測試的問題。
Parallel Appium Library 應該是 Robot Framework 的 keyword library	解決 Appium 原本透過 python 等程式語言，撰寫測試腳本的難度較高的問題。
Parallel Appium Library 應要支援平行測試	解決 Appium 與 Robot Framework –Appium Library 不支援平行測試的問題。

## 第四章 系統架構分析與實作

本章將說明 Parallel Appium Library 的系統架構，並依據第三章所提出的改善之方法進行實作。

### 4.1 Parallel Appium Library 架構設計

本函式庫分為兩大部分，分別為 AppiumInstance 與 AppiumInstance 管理。專案環境需求如表 4.1，架構設計如圖 4.1。

表 4.1 Parallel Appium Lib 環境需求

軟體名稱	下載連結
Python2.7	<a href="https://www.python.org/">https://www.python.org/</a>
Pycharm	<a href="https://www.jetbrains.com/pycharm/">https://www.jetbrains.com/pycharm/</a>
Xcode	<a href="https://itunes.apple.com/tw/app/xcode/id497799835?mt=12">https://itunes.apple.com/tw/app/xcode/id497799835?mt=12</a>
Robot Framework	<a href="http://robotframework.org/">http://robotframework.org/</a>
Node.js	<a href="https://nodejs.org/en/">https://nodejs.org/en/</a>
JAVA	<a href="http://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html">http://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html</a>
Android studio	<a href="https://developer.android.com/studio/">https://developer.android.com/studio/</a>



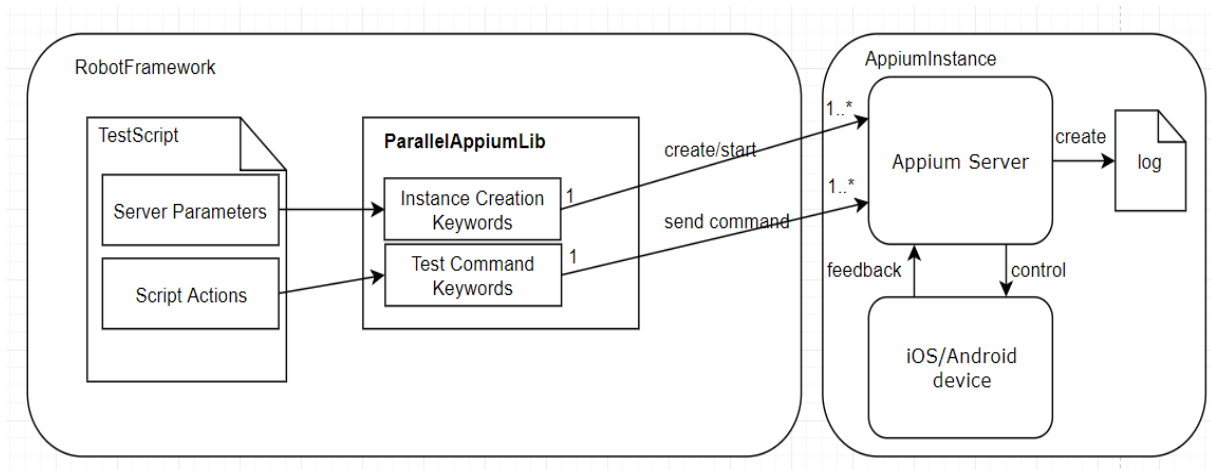


圖 4.1 Parallel Appium Lib 架構設計圖

概念上，ParallelAppiumLib 將 Appium 伺服器、手機與客戶端組合視為一體，稱為 AppiumInstance。在測試運行的時候，ParallelAppiumLib 會依照使用者輸入的參數，同時生成並管理多個 AppiumInstance，同時透過 AppiumInstance 中實踐 Appium webdriver 的指令，讓使用者能夠使用搜尋、點擊、輸入、滑動、等待等功能的 keyword。

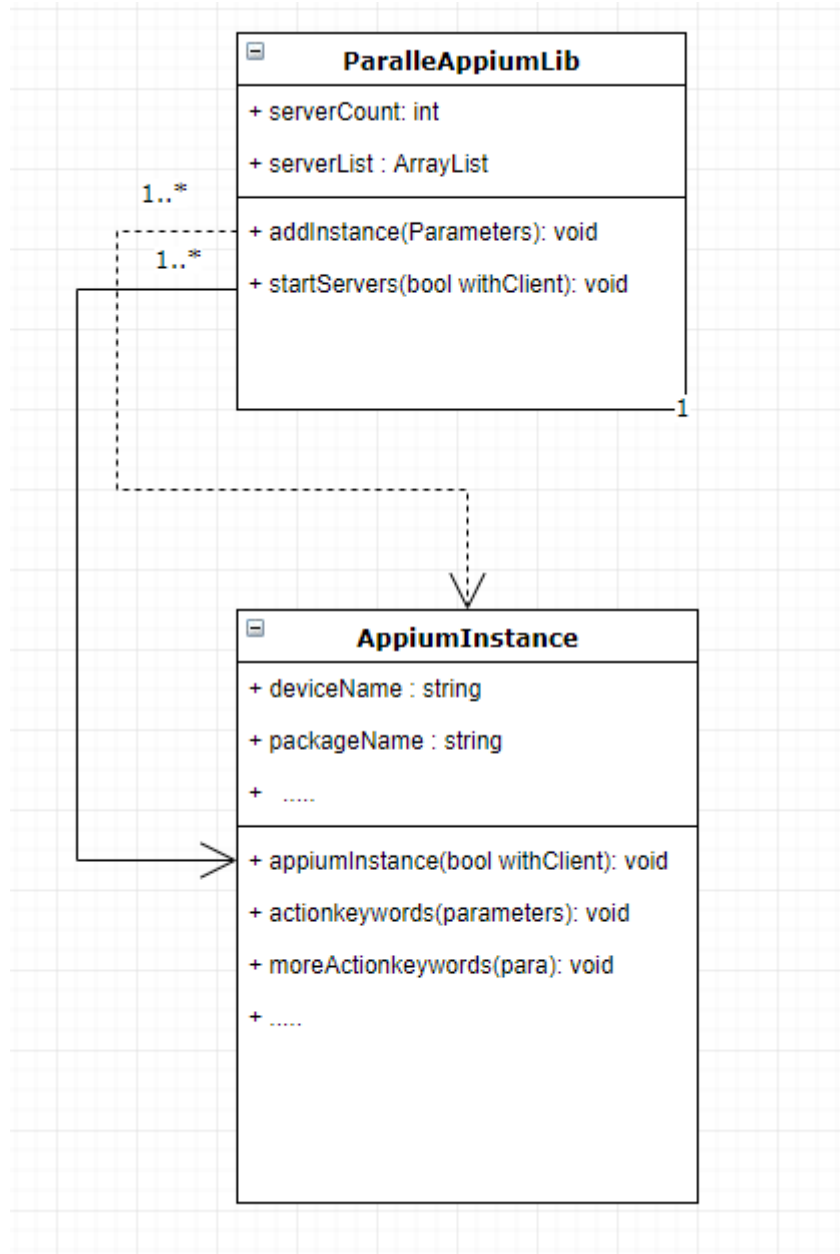


圖 4.2 ParallelAppiumLib class diagram (簡化)

## 4.2 AppiumInstance 管理

### 4.2.1 參數設定

在 ParallelAppiumLib 中我們用一個 ArrayList 來儲存多個 AppiumInstance。透過 keyword “setServerCount”，我們可以設定這個 ArrayList 在這個腳本之中所需要運行的 AppiumInstance 的數量。

“setServerCount” 使用語法：

**setServerCount** <server count>

接下來透過 keyword “addInstance”，我們可以在 ArrayList 中新增並設定一個 AppiumInstance。

“addInstance” 使用語法：

**addInstance** <device UUID> <Appium port> <bootstrapPort> <platform name>  
<platformVersion> <package name> <activity name> <browser name (for browser testing)>  
<Xcode organization ID (for iOS)> <bundleID (for iOS)>

addInstance 後面的參數是啟動 Appium 伺服器與客戶端必要的參數，依序為 UUID、appiumPort、bootstrapPort、platformName、platformVersion、packageName、activityName，以及三個 iOS 用參數 browserName、xcodeOrgID、bundleID。

最後，在設定完欲測手機的數量，並分別設定 AppiumInstance 的必要參數後，使用者必須透過 startServers 平行啟動啟動之前幾部設定的伺服器。

“startServers” 使用語法：

startServers <Boolean withClient>

若 withClient 為 true，addInstance 會再啟動的時候一併啟動 Appium 客戶端，若為 false，則只會啟動伺服器。

```
setServerCount 2
addInstance 607dd2124bfe2b5f95929539bf41e68265009de7 4020 4021 iOS 10.3.3 com.acetesting.deluge.Chat-SDK .MainActivity
addInstance EP7337JFDK 4030 4031 Android 4.3 co.chatsdk.android.app co.chatsdk.ui.login.LoginActivity
startServers true
sleep 10s
```

圖 4.3 設定並運行兩個伺服器的範例

所有的指令都會依照 AppiumInstance 在 ArrayList 中的所在位置區別，ArrayList 中的位置式所有測試指令的第一參數。如：

click\_element\_by\_accessibility\_id 0 Send

其中第一個參數「0」，代表這個指令會送往 ArrayList 中的第一個 AppiumInstance。

## 4.3 AppiumInstance

在 ParallelAppiumLib 中，AppiumInstance 是 Appium 伺服器、客戶端與手機的合體，在此章節會介紹 AppiumInstance 如何處理並運用來自 addInstance 的資料與測試制應如何運作。

### 4.3.1 AppiumInstance 參數與參數解析

由於 desired capabilities 與啟動伺服器用參數有許多重疊的地方，本專案能自行解析來自 addInstance 後面的參數，並將他分為啟動 Appium 伺服器用，以及客戶端發送給伺服器的 Desired Capabilities，分別在內部啟動伺服器與客戶端的時後使用兩者皆需要的資訊。下表是這些參數的用途。

表 4.2 addInstance 參數內部用圖表

參數名	內部用途
UUID	[伺服器參數][desired capabilities] 設定欲測手機之 UUID
appiumPort	[伺服器參數] 分配 Appium 伺服器所用的 port  [desired capabilities] 設定 client 端指令傳送的目標 port
bootstrapPort	[伺服器參數] 分配 Appium 工具 bootstrap 所收聽的 port  [desired capabilities] 設定 client 該連的 bootstrap port
platformName	[desired capabilities] 本測試要運行的手機 OS 平台
platformVersion	[desired capabilities] OS 版本
packageName	[desired capabilities] 欲測 App 之 package 名稱
activityName	[desired capabilities] 欲測 App 之 activity 名稱
browserName	[for browser test][desired capabilities] 要測試網頁瀏覽器的名稱
xcodeOrgID	[for iOS][desired capabilities]  在 Apple 所登記的開發團隊名稱
bundleID	[for iOS][desired capabilities]  App 在 Apple 所登記的 bundleID

### 4.3.2 預設 Desired capabilities 與其他參數

Parallel Appium Library 會視使用者的輸入參數發送 desired capabilities，除了使用者輸入的 desired capabilities，在本系統內部有一些為了滿足 ParallelAppiumLib 環境而預設的 desired capabilities。如 useNewWDA、startIWDP。值得注意的是 useNewWDA，Appium 透過由 Facebook 所開發的 WebDriverAgent 進行 iOS 測試，而在 ParallelAppiumLib，每次運行測試時都重新安裝一遍 WDA 可以使得測試更穩定。

desired capabilities	原因
useNewWDA	確保每次運行的時候，手機上都會安裝最新的 WDA。
startIWDP	在進行手機網頁瀏覽器測試時，為了使 ios_webkit_debug 能自動提取 webviews 資料 on iOS.

另外。為了方便測試運行，所有 Appium 伺服器都會在 localhost 上執行。

### 4.3.3 測試指令 keywords

為了達成平行測試的效果，每個 AppiumInstance 都需要分開套用、引用 Appium webdriver。透過分裝 python-client 到不同的 AppiumInstance 中，我們可以製作支援平行測試的測試指令 keyword。

在這個章節中我們會舉出關鍵的 keyword，說明用途，並解釋他是如何完成的。

### 4.3.3.1 Find Element

尋找物件是所有手機測試工具最基本的功能，ParallelAppium 也不惶多讓。在本專案中，AppiumInstance 透過 webdriver 中的 find\_element 系列函式完成所有搜尋物件指令的 keyword。

如 keyword **find\_element\_by\_accessibility\_id**，的內部是 AppiumInstance 引用 webdriver 中 find\_element\_by\_accessibility\_id(id)所做。

表 4.3 FindElement 系列 keyword

keyword	用途	範例
find_element_by_accessibility_id	透過 accessibility_id 來尋找物件	find_element_by_accessibility_id 0 ID
find_element_by_id	透過 id 來尋找物件	find_element_by_id 0 ID
find_element_by_xpath	透過 xpath 來尋找物件	find_element_by_xpath 0 //XCUIElementTypeButton[@name="Go"]
find_element_by_class_name	透過 class_name 來尋找物件	find_element_by_class_name 0 button
find_element_by_name	透過 name 來尋找物件	find_element_by_name 0 name

### 4.3.3.2 Click(Tap)

ParallelAppiumLib 中，點擊是透過組合 find 與 click 兩個動作而成。AppiumInstance 引用的 webdriver 的 Find element 系列來進行目標定位，找到後再運用函式點擊找到的物件。如 keyword **click\_element\_by\_id** 便是運用 webdriver 的 find\_element\_by\_id 定位目標，再透過.click()函式執行點擊。下表為本專案提供的點擊指令與他的範例。

表 4.4 Click 系列 keyword

keyword	用途	範例
<b>click_element_by_accessibility_id</b>	透過 accessibility_id 來點擊物件	click_element_by_accessibility_id 1 Message option button
<b>click_element_by_id</b>	透過 id 來點擊物件	click_element_by_id 0 co.chatsdk.android.app:id/chat_sdk_et_username
<b>click_element_by_xpath</b>	透過 xpath 來點擊物件	click_element_by_xpath 0 //XCUIElementTypeCell[@name="Camera Roll"]
<b>click_element_by_class_name</b>	透過 class_name 來點擊物件	click_element_by_class_name 0 button
<b>click_element_by_name</b>	透過 name 來點擊物件	click_element_by_name 0 name



tap_coordinate	透過座標來 點擊	tap_coordinate      1      670      960 10
----------------	-------------	---

### 4.3.3.3 Input Text

ParallelAppiumLib 中，點擊是透過組合 find 與 sendkey(string)兩個動作而成。再引用 webdriver 的 find 之後，用對選取目標下 sendkey 系列函式指令。Input\_text 系列 keyword 的第一個參數是伺服器編號，第二個是搜尋物件的條件資料，第三個是欲輸入的字串。

表 4.5 sendKey 系列 keyword

keyword	用途	範例
input_text_by_accessibility_id	透過 accessibility_id 來 輸入文字	input_text_by_accessibility_id 0      accesiblility_id texts
input_text_by_id	透過 id 來輸入文 字	input_text_by_id      1 identifierId      acetesting.b
input_text_by_xpath	透過 xpath 來輸 入文字	input_text_by_xpath      0 //*[@id='cmcsubj'] testmail
input_text_by_class_name	透過 class_name 來輸入文字	input_text_by_class_name 0      classname      texts
input_text_by_name	透過 name 來輸入 文字	input_text_by__name      0 name      texts

#### 4.3.3.4 Wait for condition

Wait\_until 系列 keyword 與前面幾個比起來比較不同，我們運用 webdriver 的 expected\_conditions 設定我們要等待的條件，如 ID、xpath 等，再用 WebDriverWait，等待 expected\_conditions 中的條件達成，也就是透過參數找到物件。Wait\_until 系列 keyword 的第一個參數是 AppiumInstance 編號，第二個是等待條件搜尋方法，第三個是 timeout 時間，單位為秒，timeout 時間欲設為 20 秒。

表 4.6 wait 系列 keyword

keyword	用途	範例
wait_until_page_contains_accessibility_id	以 accessibility_id 為尋找與等待的條件	wait_until_page_contains_accessibility_id 0 accessibility_id 2000
wait_until_page_contains_id	以 id 為尋找與等待的條件	wait_until_page_contains_id 0 ID 2000
wait_until_page_contains_xpath	以 xpath 為尋找與等待的條件	wait_until_page_contains_xpath 0 xpath 2000
wait_until_page_contains_class_name	以 class_name 為尋找與等待的條件	wait_until_page_contains_class_name 0 class_name 2000

wait_until_page_contains_name	以 name 為 尋找與等待 的條件	wait_until_page_contains_name 0      name      2000
-------------------------------	--------------------------	--

#### 4.3.4 Parallel Appium Lib 使用實例

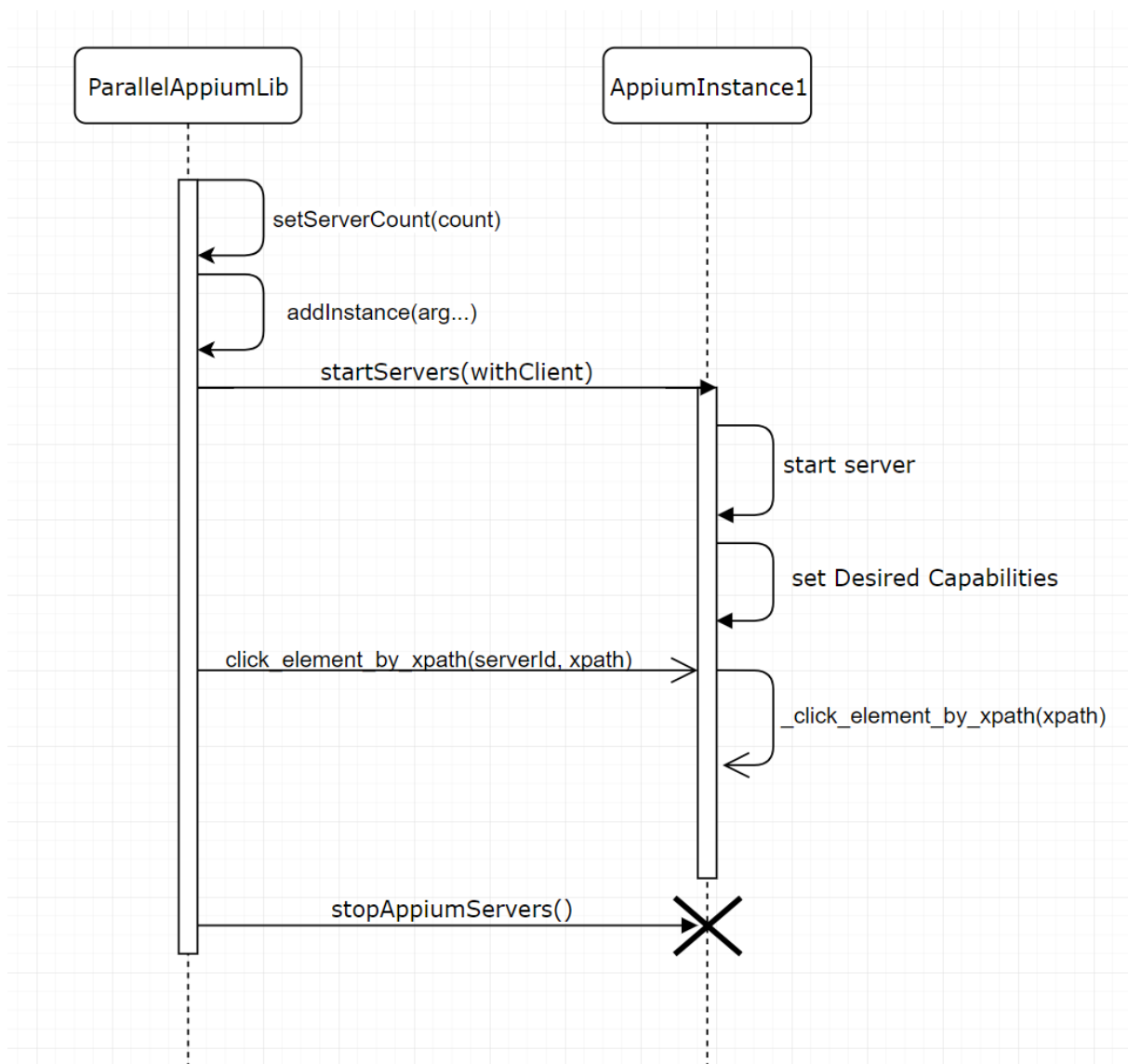


圖 4.4 以 find\_element\_by\_accessibility\_id 為例， AppiumInstance 運行測試的  
Sequence Diagram

本章節會運用簡單的範例介紹 Parallel Appium Lib，在平行互動測試彈出式訊息情境下的使用流程與運作原理。

第一步是視這次測試腳本所需之裝置數量運用 **setServerCount** 設定 AppiumInstance 的數量。第二步 用 **addInstance** keyword 依序設定每個裝置/伺服器，第三步，運用 **startServers** 啟動之前新增與設定的 AppiumInstance，完成以上步驟即可開始運行測試動作。在完成所有測試之後，運用 **stopAppiumServers** 關閉所有在背景的 Appium 伺服器。

ParallelAppiumLib 測試聊天情境腳本受到 HTTP request 的啟發。一般而言，送出 HTTPrequest 的時候，發送端在發送訊息後會等待對方的回應，若成功則繼續，若時間內沒收到訊息則算 Timeout。在我們人類使用手機聊天的時候也有類似的情形，我們在發送訊息之後會等待對方的回應，視對方的回應再決定該回甚麼訊息。在 ParallelAppiumLib 中也是如此，在 A 手機向 B 手機送出訊息後，A 手機在腳本上應該用 wait 系列指令，等待 B 手機回覆的訊息或其他指標來判斷測試的成功與否。

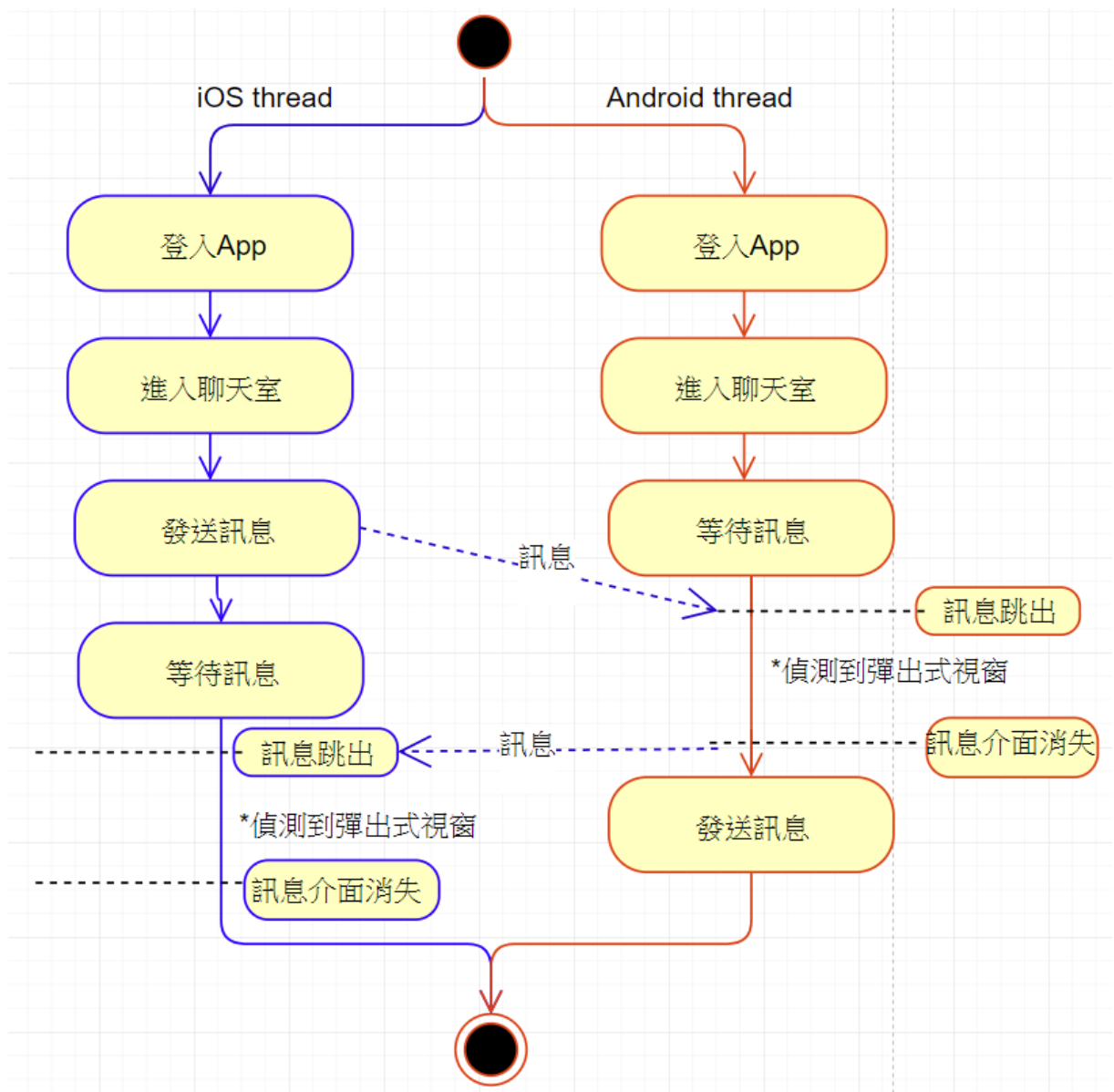


圖 4.5 ParallelAppium 兩隻手機聊天測試情境流程圖

## 第五章 比較

本章會比較 Robot Framework-Appium Library 與 Robot Framework-ParallelAppium Lib 之間功能與執行測試方式上的差異。比較原始 Appium 與 Robot Framework-ParallelAppium Lib 在撰寫測試腳本上的差異。首先會描述比較內容、接著說明比較環境，最後講解測試進行的方式，以及針對第三章需求分析的項目進行評估。

### 5.1 比較內容

Robot Framework-ParallelAppium Lib 是根據第三章所分析的內容而開發的工具，因此本論文比較的內容有三：其一，比較 Robot Framework-Appium Library 與 Robot Framework-ParallelAppium 針對伺服器與客戶端的控制功能。其二，比較原版 Appium 與 ParallelAppiumLib 撰寫的難易度。

### 5.2 環境介紹

表 5.1 電腦環境

電腦名稱	MACbook pro 2015
作業系統	OSX
CPU	Intel® Core™ i7-2600 CPU @ 3.40GHz 3.40GHz
Ram	16GB

表 5.2 手機環境

手機名稱	Sony Xperia Ultra	iPhone 5
作業系統	Android 4.3	iOS 10.3.3
CPU	Qualcomm Snapdragon 800 2.2GHz	Apple A6, 1GHz
Ram	2GB	1GB

螢幕解析度	1920 x 1080 像素	1136 x 640 像素，326 ppi 解析度
-------	----------------	---------------------------

## 5.3 功能比較

### 5.3.1 Appium Library 功能

Appium Library 除了提供基本，點擊、輸入文字、滑動、等待狀態等 keyword 之外，還有提供 switch\_application 這個比較特別的 keyword。switch\_application 可以切換正在連接的伺服器，而在 switch\_application keyword 後的測試腳本，會變成對另一台伺服器下指令，控制另一台手機。

```
| ${appium1}=      | Open Application | http://localhost:4723/wd/hub | alias=MyApp1 | platformName=iOS |
| ${appium2}=      | Open Application | http://localhost:4755/wd/hub | alias=MyApp2 | platformName=iOS |
| Click Element    | sendHello       | # Executed on appium running at localhost:4755 |
| Switch Application | ${appium1}      | # Switch using index         |
| Click Element    | ackHello        | # Executed on appium running at localhost:4723 |
| Switch Application | MyApp2          | # Switch using alias         |
```

圖 5.1 switch\_application 的使用範例

在我們的目標情境之中，他可以單執行續的在多支手機間切換，達到類似聊天的效果。但是 Appium Library 並不支援多執行續，若在多執行續下執行，每個執行續下控制的手機、伺服器會互相衝突，無法在多執行續下執行多支手機的平行測試。

另外，在單執行續下互相切換，在控制一隻手機的同時，我們便無法即時追蹤另一支手機，無法測試切換過程之中發生的事件，也無法偵測上一支手機上發生的事件。任何在切換手機的過程中開始又結束的狀態，我們無法測試。

同時，Appium Library 也只是 Appium 的客戶端，不提供啟動伺服器的功能。

### 5.3.2 ParallelAppiumLib 功能

為了改善 Appium Library 無法平行運行多支手機與不提供啟動伺服器的問題，ParallelAppiumLib 運用 ArrayList 的結構管理 AppiumInstance，每個伺服器/手機/客戶端一組，而每一組之間不互相衝突，而且可以在 ArrayList 中平行啟動。

因此便能同時追蹤多個手機，避免時間空檔中失去測試瞬發情境的機會，以及無法追蹤上一台手機的改變。同時，平行化增加同時能運行的腳本數量，縮短所有測試執行的總長，增加整體測試的效率。

### 5.3.3 比較結果

ParallelAppiumLib 不只有開啟多個伺服器的功能，而且還能在多執行緒的狀態之下分別下指令、執行測試腳本。使我們可以透過一套 keyword library，啟動並運用多台 Appium 伺服器。

## 5.4 腳本撰寫方式

### 5.4.1 原始 Appium

Appium python-client 基本上就是運用 python 程式碼呼叫 library 中的函式來向伺服器發送指令，運行測試，並沒有啟動伺服器的能力，伺服器需要依照 3.1.2 描述另行啟動、管理，是個分開的系統。以下是 Appium python-client 的測試範例。

首先必須設定 desired capabilities：



```
# Android environment
import unittest
from appium import webdriver

desired_caps = {}
desired_caps['platformName'] = 'Android'
desired_caps['platformVersion'] = '4.2'
desired_caps['deviceName'] = 'Android Emulator'
desired_caps['app'] = PATH('../..../apps/selendroid-test-app.apk')

self.driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
```

```
# iOS environment
import unittest
from appium import webdriver

desired_caps = {}
desired_caps['platformName'] = 'iOS'
desired_caps['platformVersion'] = '7.1'
desired_caps['deviceName'] = 'iPhone Simulator'
desired_caps['app'] = PATH('../..../apps/UICatalog.app.zip')

self.driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)
```

圖 5.2 透過 Python-client 設定 desired\_capabilities

接下來才能運行測試動作：

```
el = self.driver.find_element_by_accessibility_id('Animation')
self.assertIsNotNone(el)
```

```
els = self.driver.find_elements_by_accessibility_id('Animation')
self.assertIsInstance(els, list)
```

圖 5.3 透過 python-client 尋找物件[30]

Appium python-client 其實就是 python 程式碼，程式碼的學習門檻比較高。

## 5.4.2 ParallelAppiumLib 撰寫方式

ParallelAppiumLib，在 Robot Framework keyword-driven 的架構之下，所有測試指令都由 keyword 封裝。以下是 ParallelAppiumLib 撰寫範例：

```
sleep    10s
click_element_by_id    1    chat.rocket.android:id/drawee_avatar
click_element_by_id    1    chat.rocket.android:id/userStatus
sleep    10s
input_text_by_id    1    chat.rocket.android:id/editor    hi
click_element_by_id    1    chat.rocket.android:id/button_send
```

圖 5.4 ParallelAppiumLib 的使用範例

其中 keyword 後的第一個參數是伺服器在 ArrayList 中的位置。

## 5.4.3 比較結果

很明顯的，由 keyword 組成的 ParallelAppiumLib 比較淺顯易懂，相較 python-client 使用 python 程式碼，使用者不需要學習 python 程式碼的語法，也不需要很多撰寫程式碼的經驗，學習門檻較低。

## 第六章 結論與未來研究方向

### 6.1 結論

Appium 支援多個平台的特性，使得開發者在當今手機 App 的生態中更容易測試、維護自己的產品，然而 Appium 一個伺服器只能測試一隻手機的設計，使得自身無法運行平行測試，加上 Appium-python-client 是由 python 程式語言撰寫，增加學習及使用難度。

另外，Robot Framework-Appium Library 雖然能夠用 Robot Framework keyword-driven 的特性，降低學習門檻，但是不支援多執行續平行測試，無法精準的模擬多機互動的使用情境。

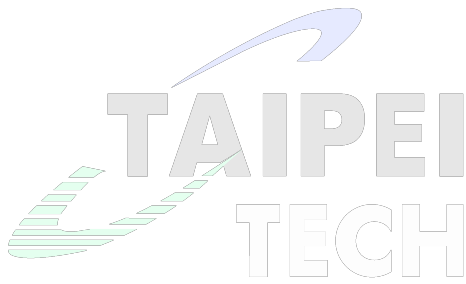
而 ParallelAppiumLib，針對這些問題提出了解決方案。

1. 透過一套 library，我們可以同時啟動並分別控制多個伺服器、客戶端的組合。
2. 可以平行控制多支手機，測試多機互動的情境。
3. 可以進行平行測試，縮短測試時間。
4. 可以驗證彈出式即時訊息。
5. Keyword driven 架構使得測試腳本能夠更簡單，學習門檻降低。

### 6.2 未來展望

1. 目前 ParallelAppiumLib 並未封裝所有 python-client 所提供之函式，限制了 ParallelAppiumLib 能夠執行的測試動作。希望未來能補上更多的 keyword，使得測試動作更加健全。
2. 因為 ParallelAppiumLib 中使用了本論文中固定的 desired capabilities，使用者無法發送其他的 desired capabilities，限制了 Appium 的設定範圍。未來需要改變這點，使使用者能夠輸入更多其他 desired capabilities。

3. 目前 ParallelAppiumLib，並未支援所有 Appium 所有的伺服器參數，未來希望能結合第二點，使使用者能夠更自由的設定 Appium 伺服器。



## 參考文獻

- [1] iOS, Available: <https://www.apple.com/tw/ios>
- [2] Android, Available: <https://www.android.com/>
- [3] Janzen, D.S., Saiedian, H., “Does Test-Driven Development Really Improve Software Design Quality?” Software, IEEE, vol. 25, March-April 2008
- [4] Mark Fewster, Dorothy Graham, “Software test automation: effective use of test execution tools” Addison Wesley, 1999.
- [5] Acceptance Test, Available: [https://en.wikipedia.org/wiki/Acceptance\\_testing](https://en.wikipedia.org/wiki/Acceptance_testing)
- [6] Appium, Available: <http://appium.io/>
- [7] Native App, Available: <https://developer.android.com/studio/command-line/adb.html>
- [8] Hybrid App, Available: <https://developer.android.com/studio/command-line/adb.html>
- [9] Mobile Web App, Available: <https://developer.android.com/studio/command-line/adb.html>
- [10] Robot Framework, Available: <http://robotframework.org/>
- [11] 劉展君，基於 Robot Framework 的平行處理測試函式庫：以手機測試為例，碩士論文，國立臺北科技大學資訊工程系碩士班，台北市，2014
- [12] XCUITest, Available <https://developer.apple.com/reference/xctest>
- [13] UIAutomation, Available: <https://developer.apple.com/library/ios/documentation/DeveloperTools/Reference/UIAutomationRef/>
- [14] UIAutomator, Available: <https://developer.android.com/topic/libraries/testing-support-library/#UIAutomator>
- [15] Selendroid, Available: <http://selendroid.io/>
- [16] WinAppDriver, Available: <http://github.com/microsoft/winappdriver>
- [17] RESTful, Available: [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)
- [18] Desired Capabilities <https://appium.io/docs/en/writing-running-appium/caps/>
- [19] JSON, Available: <https://zh.wikipedia.org/wiki/JSON>
- [20] 圖片參考：[https://nishantverma.gitbooks.io/appium-for-android/appium/why\\_appium.html](https://nishantverma.gitbooks.io/appium-for-android/appium/why_appium.html)
- [21] Ken Pugh, “Lean-Agile Acceptance Test-Driven Development” United States: Addison Wesley, 2011.

- [22] 圖片參考：<http://robotframework.org/#libraries>
- [23] Robot Framework -run keyword async, Available:  
<https://github.com/awadhn/robotframework-run-keyword-async>
- [24] 圖片參考：<https://github.com/awadhn/robotframework-run-keyword-async>
- [25] Robot Framework-Appium Library, Available:  
<https://github.com/serhatbolsu/robotframework-appiumlibrary>
- [26] Node.js, Available: <https://nodejs.org/en/>
- [27] Appium server arguments:  
<https://appium.io/docs/en/writing-running-appium/server-args/>
- [28] Python-client, Available: <https://github.com/appium/python-client>
- [29] JAVA-client, Available: <https://github.com/appium/java-client>
- [30] 圖片參考：<https://github.com/appium/python-client>

